Parallel Basic Software

Takashi Chikayama

First Research Department

ICOT Research Center

Parallel Basic Software in the FGCS Project

- Based on five models of parallel inference machine
- Parallel and distributed implementations of a concurrent logic programming language KL1
- Software development environment for KL1 provided by a parallel operating system PIMOS

→ Proven thru experimental // application systems

Problems of the Parallel Inference System

- Built on special purpose hardware not widely available
- KL1 as the only language
  → Hard to utilize existing software
- Alien interface → high initial threshold to get over
- Lack of efficient higher level language implementations

⇒ Obstacles to broader utilization

Parallel Basic Software in the Follow-on Project

- Built on systems available in the market
- Linkage with programs in other languages
- User interface consistent with Unix
- Higher level features by tuned-up theorem provers

⇒ Wider availability and lower initial introduction cost

Evaluation of PIM

- One bit reference count in pointers (MRB)
  - Small management cost with slight HW support
  - Destructive updates of arrays → random access
  - Incremental GC costs high in free list maintenance
- Shared memory parallel implementation
  - Locking is not so costly; simple compare & swap do
  - Automatic load balancing works fairly
  - GC of shared memory is costly due to bus bottleneck

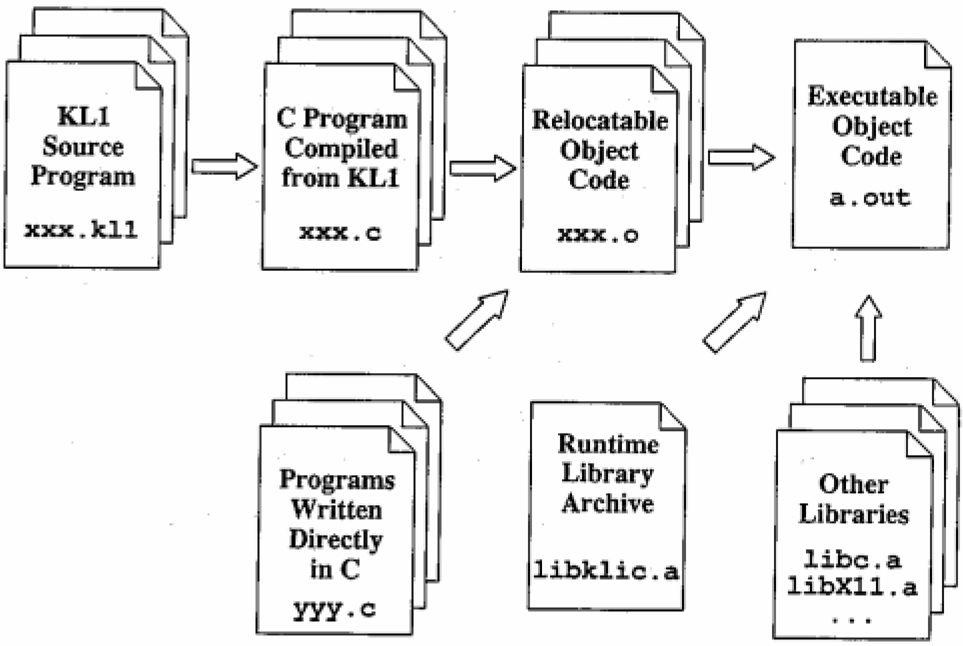
Evaluation of PIM (continued)

- Distributed memory parallel implementation
  - Two-level addressing allows local garbage collection
  - Weighted reference counting global GC works fine
  - Lazy data transfer works but its overhead is large
- A reasonably efficient implementation as a whole, with some room for further optimization

KLIC: A Portable KL1 Implementation

- C as an intermediate language → Portability!
- Modular design → Portability!
- Collection of smaller programs and libraries
  ⟺ Single integrated environment for everything
- Smooth interface to programs in other languages

Unix-Style Compilation



Basic Design of KLIC

- Only two tag bits in pointers; no MRB
- "Generic objects" for built-in type variety
- Single core implementation for all variations
  (debugging/production, sequential/parallel)
  The same code linked with different runtime libraries

Generic Objects

A unified framework for system extension

- Object-oriented interface with the core implementation
  - Manipulation through "generic methods" only
  - Physical representation encapsulated
- Object-oriented foreign language interface
  - Foreign language data in KLIC heap
  - Object migration by defining message encoding

Three Kinds of Generic Objects

Data objects for immutable data

- Explicit manipulation thru method invocation
- For more builtin types, &c

Consumer objects for data-driven processes

- Associated with an uninstantiated variable
- Activated on variable instantiation

Generator objects for demand-driven processes

- Associated with an uninstantiated variable
- Activated on demand of variable value

Sequential Performance

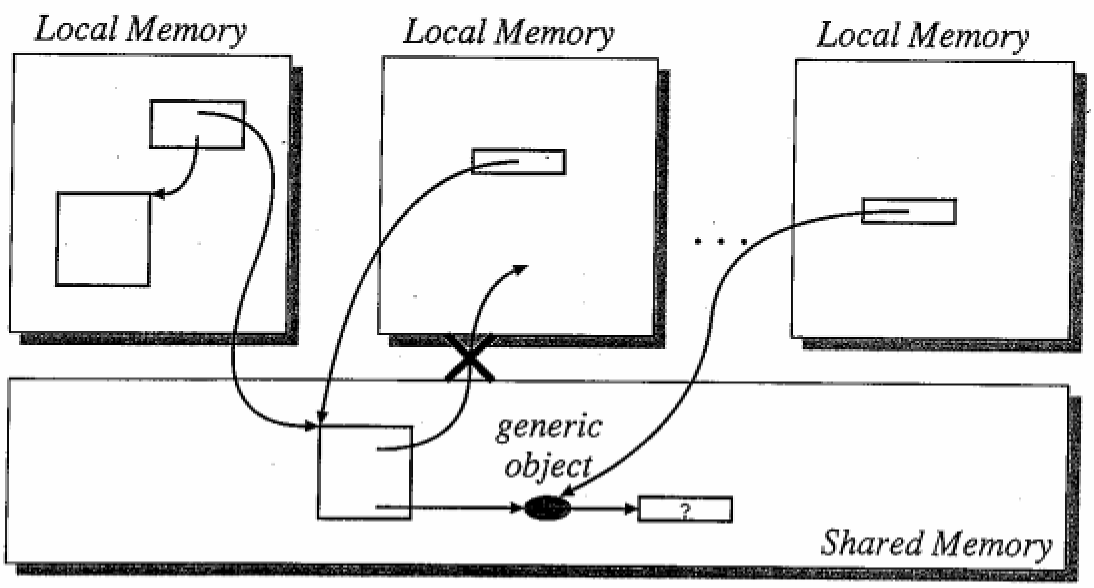
- Twice as fast as SICStus Prolog native code both for small benchmarks and the KLIC compiler
- Smaller code size than SICStus native code
- KLIC on PIM shows similar performance as its original KL1 implementation

= Single processor performance of // implementations

KLIC: Shared Memory // Implementations

- Local heaps + a shared heap
  - Pointers from local heap to shared heap
  - No pointers from shared heap to local heap
  - Local data copied to shared heap when necessary
- Shared variables as generic objects
  (Locking and data copy needed)
- Independent & asynchronous GC on local heap
  Bus bottleneck removed

Local Heaps and a Shared Heap



KLIC: Distributed Memory // Implementation

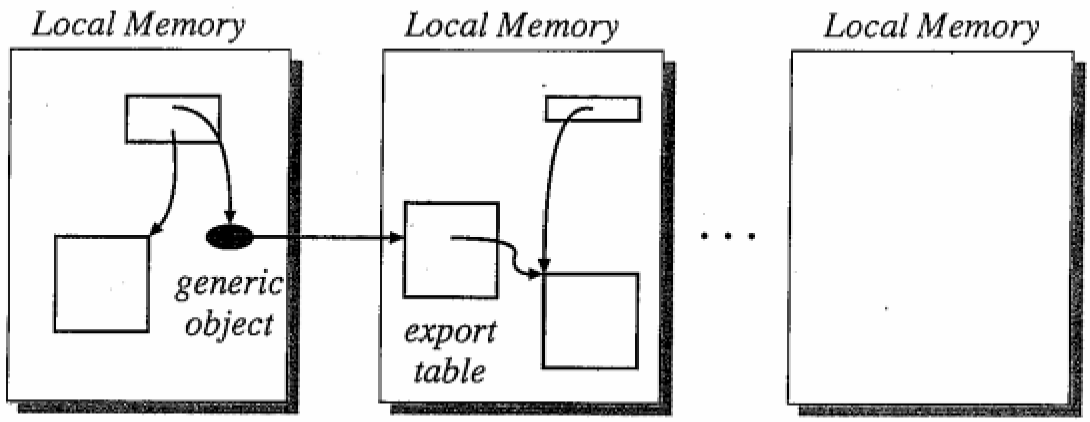
Inherits the schemes of the implementation on PIMs

- Two level addressing (local and global addresses)
- Local GC enabled by export table maintenance
- Global GC thru weighted reference counting

No need to support an operating system

⇒ Simplified than the PIM implementation

Interprocessor Reference Management



Portability

- Sequential core implementation
  workstations, servers and PCs (DOS, OS2, Linux)
- Shared memory // implementations on Sun and DEC
- Distributed memory // implementations
  - On message passing libraries: PVM, MPI
  - On system-specific features: active messages &c
  - Using shared memory as message passing media

Pool of PIMOS

- Table maintenance utility for KL1 programs
- Used extensively in application systems on PIM
- No parallelism inside
  - Communication latency for distributed clients
  - Load concentration to the server node
  - Memory usage imbalance

  Worse on KLIC on conventional hardware

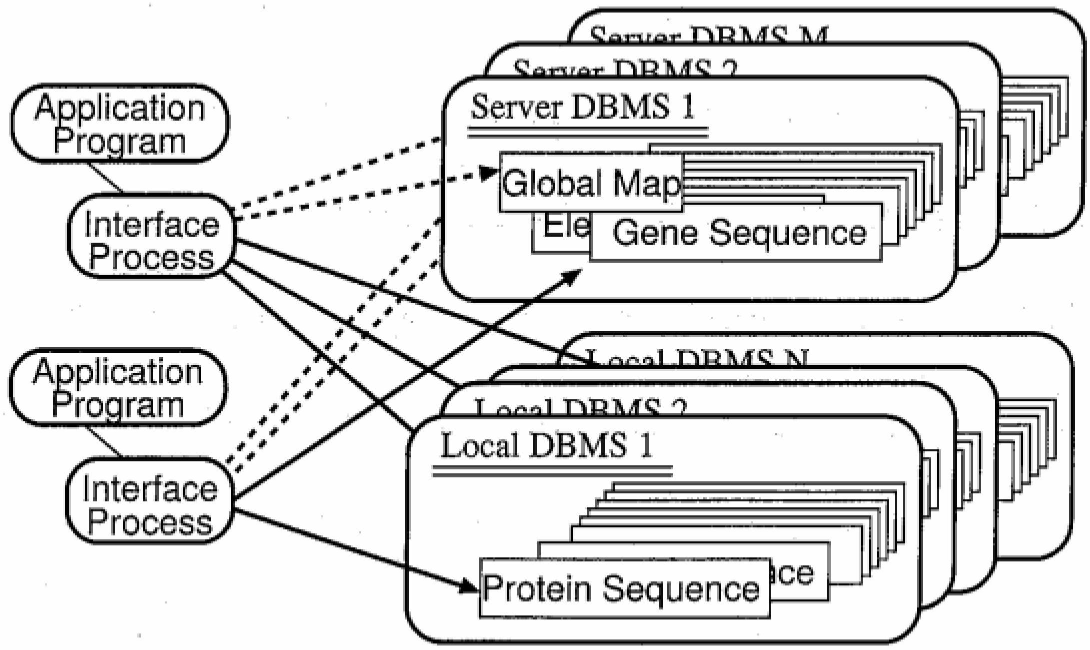
Distributed Pool

- Caching copies of recently accessed data
  - Higher access locality
  - Better load distribution
- Distributing data on their keys
  - Better utilization of distributed memory
- Coherence control by asynchronous message exchange
  $\longrightarrow$ New protocol to maintain cache coherence

Kappa: A Parallel DBMS

- Data management on a nested relational model
  $\longrightarrow$ More efficient handling of complex structured data
- Integration of distributed DBMSs running in parallel
  - Speed-up by parallel processing
  - Single integrated view from applications
- Lower level processing in C
  $\longrightarrow$ Performance approaching conventional DBMS

An Example Configuration of Kappa



MGTP: A Bottom-Up Theorem Prover

- A prover for full first-order logic
- Proof by generating models for the given axiom set
- Almost linear speed-up by parallel model generation
- Solved an open problem in quasi-group theory
- Was inefficient for a certain problem classes

MGTP: R&D in FGCS Follow-on Project

- Non-Horn magic set, to specify top-down proof control
- Constraint MGTP, for efficient constraint propagation
- Translation of modal logic to a form MGTP can handle
- Distributed MGTP, for slower communication media

⇒ A general tool for building knowledge based systems

Conclusion

- KLIC formed a basis for wider utilization of FGCS technologies
- Application software systems on PIM are now available on widely available computer systems
- More room remains for optimization and refinements